OpenTWIN 2,4 GHz User Data Protocol Specification with Terminal Extensions

rev. 1.2, Petr Sladek, 4th October 2010

1. Introduction

MZK Servis Czech Republic realized that compatibility of radio control over ISM 2,4 GHz band is the most important for all customers. User Data Specification covers user data interface and transport between TXC and RXC modules. OpenTWIN uses the IEEE 802.15.4 radio physical layer.

2. Physical Layer

Both the TXC and RXC modules are equipped with UART, internally connected RxD and TxD signals.

UART Voltage Levels:

0 – 0.5 V: Logic Low

2.8 – 3.5 V: Logic High

Output sink/source current: max. 1.5 mA

Output protected against short-circuit by 220R internal resistor.

UART is operated at 19 200 bps, 8 data bits, even parity, 2 stop bits.

Maximum effective over-air speed TXC to RXC: 960bps (120bytes/s).

Maximum effective over-air speed RXC to TXC: 3200bps (400bytes/s).

Note that 3200bps is a maximum speed and will be lower for RXC to TXC in worse conditions because data transfer is secured (like TCP connection).

Pin out locations:

TWIN5, TWIN6: UART Not implemented, only internal service (SOP 0x00).

TWIN7: UART shared with channel 7.

TWIN8: UART shared with channel 8.

TWIN10: UART dedicated JR connector.

3. Media Access Control Layer

Both the TXC and RXC modules are passive operated. UART receive state is default to prevent collisions with UART transmitter. User have to ask the TXC or RXC if any data are in rx/tx buffers.

User have to send every approx 20ms query [UFC_Q] (0xFF) to TXC or RXC if data are in buffers.

TXC answer (no data available): [UFC_E]

TXC answer (data available): [data0][data1]..[dataN] [UFC_E]

RXC answer (no data available): [UFC_E]

RXC answer (no data available, input buffer full): [UFC_F]

TXC answer (data available): [data0][data1]..[dataN] [UFC_E]

UFC_E code (0xFE).

UFC_F code (0xFD).

The UFC_ codes are only media-access control and are not passed into data stream. All UFC_ codes are reserved and user have to avoid such data in user packets.

TXC has UART incoming buffer length 32bytes (UART to TXC). Outgoing buffer 64bytes (Air to TXC).

RXC has UART incoming buffer length 64bytes (UART to RXC). Outgoing buffer 32bytes (Air to RXC).

Tx/Rx Reaction Time Requirements:

Device have to control the UART using IRQ or by using very fast polling. Tx/Rx control of UART TX and RX module should be within 100us.

Required Delay after UFC: Considering RXC or TXC sends data with two stop bits and some USART interfaces (AVR) does not support two stop bits on reception (RXD) user have to wait one character (>53us) before it will starts sending data. Otherwise collision on START bit occurs.

Implementation note!: Any device connected to RXC or TXC module have to clean-up the UART data stream from UFC MAC data. Realize that packet 0x00 0x31 0x41 0xF0 will be transmitted from RXC UART filled with UFC_E or UFC_F randomly as you match the radio upstream:

Example: MZK Terminal sends: 0x00 0x31 0x41 0xF0 (SOP,...data,...EOP)

Note: [UFC_Q] is sent by sensor (0xFF).

Two fractions of packet (UART tap):

RXC sends after UFC_Q query: [UFC_Q] 0x00 0x31 0xFE [UFC_Q] 0x41 0xF0 0xFE

No fractions:

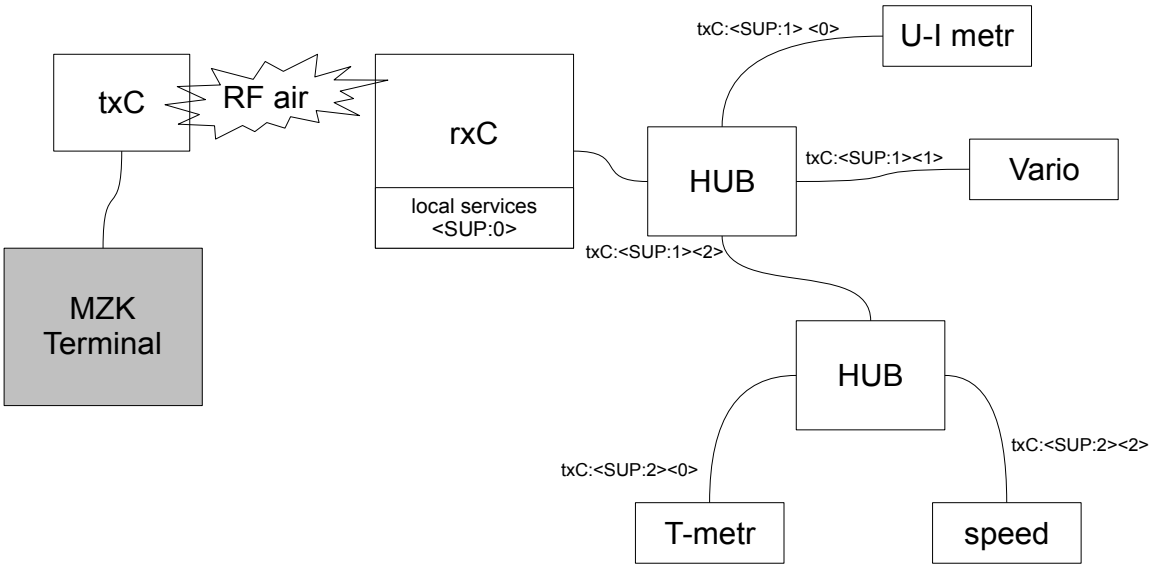
or RXC sends after UFC_Q query: [UFC_Q] 0x00 0x31 0x41 0xF0 0xFE

After your UART RXrdy interrupt routine clean-up the UART data you will get 0x00 0x31 0x41 0xF0.

4. Transport Layer

TXC to RXC data stream is not connected i.e. data are transmitted but not checked for reception in RXC. This is similar to UDP. Typically dedicated to query sensors in the “air”. Bytes are transmitted correctly (CRC) but some can be missing. Requires some packet-length checking plus headers.

RXC to TXC data stream is connected i.e. data are transmitted and checked for reception in TXC. This is similar to TCP. Typically dedicated to get data from sensors to “ground”.



Device addressing from the TXC to the device using the RXCs and HUBs.

Packet format:

[SOP|HUB_LEN][HUB_PORT#_LEVEL1]..[HUB_PORT#_LEVELn][user data][EOP]

SOP = 0x00

HUB_LEN = 0x00 – 0x07, length of HUB_PORT#LEVEL records.

user data: 0x08 - 0xEF

EOP = 0xF0

HUB_PORT#_LEVEL = ASCII “0”, “1”...“?” (0x30-0x3f)

| is bit-wise OR

Note that <u>any device is a HUB, even RXC</u>. Each HUB has own local port (SOP=0x00).

Enumeration recognize four types of devices: RXC, HUB, Sensor, Cascade-Sensor (chain).

<u>Examples:</u>

Local function located in RXC module such as RXC Battery Voltage:

[0x00][user_data]..[0xf0]

Sensor located in RXC UART port 0 module such as current monitor:

[0x01][“0”][user_data]..[0xf0]

Sensor located in HUB UART port 1, HUB is connected to RXC port 0:

[0x02][“0”][“1”][user_data]..[0xf0]

Addressing from the device to the TXC

Packet format from device:

[SOP=0x00][data][EOP]

HUBs and RXCs add HUB records accordingly.

Example:

Sensor located in RXC UART port 0 module such as current monitor sends data [0x00][data][0xf0].

From TXC UART port you obtain: [0x01][“0”][data][0xf0].

Local function located in RXC module such as RXC Battery Voltage:

[0x00][user_data]..[0xf0]

5. Session Layer (rev 1.2 and higher)

Version 1.0 and 1.1 not include any session layer. Session layer include one byte of data stream to add packet serialization information. Adding packet number can improve overall bandwidth because you can send several queries to sensors in one time and you don't have to wait on response. Session wrapper will align packet sent to sensors accordingly so that user see Query/Response/Timeout behaviour.

Byte is added on place “length” after the Hub record. Minimal value of length is 0x30 so we have a space 0x08-0x2F for session packet ID.

Record 0x20 – 0x2F is PACKET ID if used.

[SOP|HUB_LEN][HUB_PORT#_LEVELn][PACKET ID code][user data][EOP]

As you can see there is a possibility to send up to 16 different packets to same device in one time. Responding device have to use the same packet ID in reply packet.

Protocol 1.2 is backward compatible and is suggested for all new devices developed for OpenTWIN.

6. Terminal Extensions & Enumeration

Terminal Extensions is a set of simple commands for reception and graphical interpretation of measured data. Any device connected to MZK's Terminal have to support basic enumeration commands.

Note: all “x” are x ASCII values.

Ping: <p>

Ping is for basic enumeration process. Device is asked for presence in port.

Command “p”: [SOP&HUB_recLen][HUB records][“1”][“p”][EOP]

Reply revision 1.0 and 1.1:

Answer “Y”: [SOP&HUB_recLen][HUB records][“1”][“Y”][EOP]

Reply revision 1.2:

Answer “Y2”: [SOP&HUB_recLen][HUB records][“2”][“Y2”][EOP]

Note: examples herein are 1.0-1.1, add PACKET ID if you implement rev 1.2:

Device type: <?t>, allowed: H=hub, S=sensor, C=chain and R rxc types.

Rev 1.0-1.1:

Command “?t”: [SOP&HUB_recLen][HUB records][“2”][“?t”][EOP]

Answer hub 4 ports “H4”: [SOP&HUB_recLen][HUB records][“2”][“H4”][EOP]

Answer sensor 0 ports “S0”: “S0”

Cascade sensor 1 port “C1”: “C1”

Answer rxc 1 port “R1”: “R1”

Rev 1.2: (with PACKET ID 0x20-0x2F)

Command “?t”: [SOP&HUB_recLen][HUB records][0x20][“2”][“?t”][EOP]

Command “?t”: [SOP&HUB_recLen][HUB records][0x21][“2”][“?t”][EOP]

Answer: first packet 0x20 was lost due to noise

Answer hub 4 ports “H4”: [SOP&HUB_recLen][HUB records][0x21][“2”][“H4”][EOP]

Note: Cascade/Chain sensor is sensor with low count extra port embedded HUB (typically 1, 2 ports).

Info: <?i>

Command: “?i”

Answer: name of device

Layout: <?l> (L as layout)

Command: “?l”

flow layout:

Answer: “F“

table layout: Table, 2 Columns, 4 Rows:

Answer: “TC2R4“

Command Query: <?c>

Command Query: device available one-character commands (extends basic enumeration)

Command: “?c”

Answer: [SOP&HUB_recLen][HUB records][{length of str+0x30}][one_character_cmd_string][EOP]

Note: device answer can be “uvabc”, which means: device understands u, v, a, b, c commands as measured values, etc. Reserved characters: “p” and “x”. User cannot use reserved commands.

Note: order of characters specifies default flow layout order e.g. “abcd” will produce: A C D flow layout output on screen if A, C, D is visible and B is invisible. Visibility can be changed by user.

Value Name <#c>:

Name of value returned by command. “c” is a character of command. e.g. “v” for voltage of RXC.

Command: “#c”

Answer: name_of_value_string

Value Unit <%*c*>:

Unit of value returned by command. “c” is a character of command. E.g. Answer: “mA” for current sensor.

Command: “%*c*”

Answer: name_of_value_string

Default Value Layout Config <**c*>:

Command: “**c*”

Used for flow-layout and list layout:

default visible:
Answer: “V”

default invisible:
Answer: “I“

Used for table layout:

default visible, column 1, row 2:
Answer: “VC1R2“

default invisible, column 2, row 3:
Answer: “IC2R3“

Graphics of Value (display type) <$*c*>: *PRELIMINARY*

Graphical interpretation of value. “c” is a character of command. e.g. “v” for voltage of RXC.

Currently defined records for plain text, plain text low intensity, bar-graph and tachograph.

Command: [SOP&HUB_recLen][HUB records][“2”][“$c”][EOP]

Bar-graph answer:

Answer: [SOP&HUB_recLen][HUB records][{length of record+0x30}][bN<min,max>][EOP]

N specify refresh rate 0-9.

Example (battery available gas 0-100%):

Command: [SOF&HUB_recLen][HUB records][“2”][“$g”][EOP]

Answer: [SOF&HUB_recLen][HUB records][0x38][“b8<0,100>”][EOP]

Plaintext answer:

Answer: [SOP&HUB_recLen][HUB records][“2”][pN][EOP]

N specify refresh rate 0-9.

Plaintext low intensity:

Answer: [SOP&HUB_recLen][HUB records][“2”][lN][EOP]

N specify refresh rate 0-9.

Flag: (L/H), active H

Answer: [SOP&HUB_recLen][HUB records][“2”][fN][EOP]

N specify refresh rate 0-9.

Tacho-graph answer:

Answer: [SOP&HUB_recLen][HUB records][{length of record+0x30}][bN<min,max>][EOP]

N specify refresh rate 0-9.

Sensor settings:

Settings Query: <?@>

Settings Command Query: device available one-character command settings “@” commands.

Command: [SOP&HUB_recLen][HUB records][“2”][“?@”][EOP]

Answer: [SOP&HUB_recLen][HUB records][{length of str+0x30}][one_character_cmd_string][EOP]

All @commands have to support Value Name and Value Unit command (“#@c” and “%@c” c is command)

Settings:

Send “@c” to get value of specific setting, i.e. @m for maximum current limit.

Send commands below to manipulate with value and retrieve new value:

Send “@cR” to reset value to default

Send “@cM” to set minimum value

Send “@cX” to set maximum value

Send “@cu” (“@cd”) to increment(decrement) value by minor increment

Send “@cU” (“@cD”) to increment(decrement) value by major increment

Send “@cEnnn.mmm” to enter value nnn.mmm (ex. 123.45)

8. Release Notes

rev1.0:

First release.

rev1.1: 6st June 2010

Enumeration: Value Unit %c; New Device type: C: Cascade/Chain sensor;

Sensor Default Layouts, Default Layout configuration.

rev1.2: 4th October 2010

Implementation of session layer. Settings query/commands.